Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 00 (2014) 000–000

Procedia Engineering

www.elsevier.com/locate/procedia

23rd International Meshing Roundtable (IMR23)

Two mesh deformation methods coupled with a changing-connectivity moving mesh method for CFD applications

Nicolas Barral[a], Edward Luke[b], Frédéric Alauzet[a,*]

[a]*Gamma3 Team, INRIA Paris-Roquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153, Le Chesnay Cedex, France*
[b]*Department of Computer Science and Engineering, Box 9637, Mississippi State University, MS 39762, United States*

Abstract

Three-dimensional real-life simulations are generally unsteady and involve moving geometries. Industry is currently very far from performing such body-fitted simulations on a daily basis, mainly due to the robustness of the moving mesh algorithm and their extensive computational cost. A moving mesh algorithm coupled to local mesh optimizations has proved its efficiency in dealing with large deformation of the mesh without re-meshing. In this paper, the coupling of this algorithm with two mesh deformation techniques is studied: an elasticity PDE-based one and an explicit Inverse Distance Weighted interpolation one, and both techniques are compared. The efficiency of this method is demonstrated on challenging test cases, involving large body deformations, boundary layers and large displacements with shearing. Finally, the moving mesh algorithm is coupled to a CFD flow solver.

Peer-review under responsibility of organizing committee of the 23rd International Meshing Roundtable (IMR23).

Keywords: Moving mesh; Inverse Distance Weighted interpolation; Elasticity-based mesh deformation; ALE simulations

1. Introduction

The growing expectations of the industrial world for simulations involving moving geometries have given a boost to this research field for the last decades. Moving mesh simulations are currently used in many research fields: ballistics, biomedical, aeronautics, transports... These simulations combine the difficulties associated with unsteadiness, mesh movement and fluid-structure or structure-structure coupling, and are generally hard to perform and very costly in terms of CPU time.

Three leading methodologies have been designed in the literature to handle geometry displacements during numerical simulations: body-fitted approach [6,7,10,17,23], Chimera methods [8,9,21] and immersed/embedded boundary methods [18,22]. Each of them has its own strengths and weaknesses. In this work, we consider the first class of methods where the time-evolving computational domain is treated with a *body-fitted approach*, which means that the computational mesh follows time-evolving geometries in their movement. Only simplicial unstructured meshes are considered mainly because several fully-automatic software packages are available to generate such meshes [16,20].

* Corresponding author. Tel.: +33 1 39 63 57 93.
E-mail address: frederic.alauzet@inria.fr

Peer-review under responsibility of organizing committee of the 23rd International Meshing Roundtable (IMR23).

Another reason is that our final objective is to use highly anisotropic metric-based mesh adaptation [2,3] on moving mesh simulations, and simplicial elements still remain the most flexible to handle this technology.

Unfortunately, the fixed-connectivity constraint imposed by the classical flow solver numerical scheme Arbitrary-Lagrangian-Eulerian (ALE) framework considerably limits the efficiency of body-fitted moving mesh techniques. If the problem induces large displacements of the geometry, the mesh distortion can adversely affect the accuracy and stability of the numerical schemes and often make the ALE approach unpractical.

Two different methods have been proposed to handle large displacements moving mesh simulations. The first one consists of moving the mesh as much as possible, keeping the connectivity fixed, and solving the equations in a fully ALE manner, until the mesh becomes too bad and a global or local remeshing is performed, the solution being interpolated on the new mesh, see for instance [7]. In the case of displacements with large shearing, a large number of remeshing is performed, which is prohibitive in terms of CPU time. The second approach aims at maintaining the best possible mesh quality while moving using several local mesh optimizations such as vertex addition or collapsing and connectivity changes [10,13]. This strategy is extremely robust and maintains a good mesh quality throughout the simulation, but involves a large number of solution interpolations after each mesh modification, and the numerical method does not fully comply with the ALE framework.

A new approach for moving mesh simulations has been proposed in [1], which is compatible with the ALE framework and able to handle anisotropic adapted meshes. First, the number of solutions required for mesh deformation – achieved through an elasticity-based PDE method– is significantly reduced. Second, the *m*esh deformation algorithm is coupled with local mesh quality optimizations[12,15] *using only vertex smoothing and edge/face swapping*. This mesh-connectivity-change operator is especially powerful in handling shear and large deformation movement, since no re-meshing is required.

In this paper, further demonstration of the robustness of this algorithm is provided: we propose to use an explicit interpolation method to compute the mesh deformation, and compare this method to the elasticity-based one on new challenging cases. Finally, this algorithm is successfully coupled to a three-dimensional CFD ALE solver.

This paper begins with a reminder of the changing-connectivity moving mesh algorithm. The two methods used to compute the mesh deformation are then described and compared on some challenging examples. The paper ends with CFD applications, where an ALE solver has been coupled to the moving mesh method.

2. Mesh-connectivity-change moving mesh strategy

The strategy developed to move the inner mesh following the moving boundaries involves two main parts. First, the computation of the *mesh deformation*: inner vertices are assigned a trajectory, and thus a position for future time steps. Second, the *optimization* of the mesh, in which the positions computed in the mesh deformation phase are corrected and connectivity changes are performed to ensure a good mesh quality. This strategy has proven to be very powerful [1]: large displacement of complex geometries can be performed preserving a good mesh quality without any remeshing. Note that, in the scope of this paper, the surface meshes are not optimized (the surface vertices are not moved on the surface).

2.1. Mesh deformation phase

During the first phase of the algorithm, the displacement of all volume vertices, *i.e.,* the mesh deformation, is computed. To this end, two methods are considered in this paper: an elasticity-based PDE method or an explicit Inverse Distance Weighted (IDW) interpolation method. Both will be discussed more in depth in Section 3. In either cases, the cost of the mesh deformation solutions can be reduced by: (i) using a dedicated coarser mesh to compute the mesh deformation (ii) rigidifying some regions around tiny complex details of the geometries.

2.2. Improving mesh deformation algorithm efficiency

Mesh deformation algorithms, are known to be an expensive part of dynamic mesh simulations, as their solution is generally required at each solver time step (or each few solver time steps). To reduce the number of such solutions, we

propose to solve the mesh deformation problem for a large time frame of length Δt instead of doing it at each solver time step δt. While there is a risk of a less effective mesh displacement solution, it is a worthwhile strategy if our methodology is able to preserve the mesh quality along large displacements. Solving the mesh deformation problem once for large time frame is problematic in the case of curved or accelerated trajectories of the bodies. To enhance the mesh deformation prescription, accelerated-velocity curved, *i.e.,* high-order, vertex trajectories are computed.

The paths of inner vertices can be improved by providing a constant acceleration $\mathbf{a}$ to each vertex, which results in an accelerated and curved trajectory. During time frame $[t, t+\Delta t]$, the position and the velocity of a vertex are updated as follow:

$$\mathbf{x}(t+\delta t) = \mathbf{x}(t) + \delta t\, \mathbf{v}(t) + \frac{\delta t^2}{2}\mathbf{a} \qquad \text{and} \qquad \mathbf{v}(t+\delta t) = \mathbf{v}(t) + \delta t\, \mathbf{a}\,.$$

Prescribing a velocity and an acceleration vectors to each vertex requires solving two mesh deformation problems. If inner vertex displacement is sought for time frame $[t, t+\Delta t]$, boundary conditions are imposed by the location of the body at time $t+\Delta t/2$ and $t+\Delta t$. These locations are computed using body velocity and acceleration. Now, to define the trajectory of each vertex, the velocity and acceleration are deduced from evaluated middle and final positions:

$$\begin{aligned} \Delta t\, \mathbf{v}(t) &= -3\,\mathbf{x}(t) + 4\,\mathbf{x}(t+\Delta t/2) - \mathbf{x}(t+\Delta t) \\ \frac{\Delta t^2}{2}\mathbf{a} &= 2\,\mathbf{x}(t) - 4\,\mathbf{x}(t+\Delta t/2) + 2\mathbf{x}(t+\Delta t)\ . \end{aligned}$$

In this context, it is mandatory to certify that the mesh motion remains valid for the whole time frame $[t, t+\Delta t]$, which is done computing the sign of the volumes of the elements all along their path [1].

2.3. Local mesh optimization

It has been proposed [1] to couple mesh deformation with local mesh optimization using smoothing and generalized swapping to achieve efficiently large displacement in moving mesh applications. Connectivity changes are really effective in handling shear and removing highly skewed elements. Here, we briefly recall the mesh optimization procedure in the generalized context of metric-based mesh adaptive optimization.

For 3D adapted meshes, an element's quality is measured in terms of element's shape by the quality function:

$$Q_{\mathcal{M}}(K) = \frac{\sqrt{3}}{216}\frac{\left(\sum_{i=1}^{6} \ell^2_{\mathcal{M}}(\mathbf{e}_i)\right)^{\frac{3}{2}}}{|K|_{\mathcal{M}}} \qquad \in [1, +\infty]\ , \tag{1}$$

where $\ell_{\mathcal{M}}(\mathbf{e})$ and $|K|_{\mathcal{M}}$ are edge length and element volume in metric $\mathcal{M}$. Metric $\mathcal{M}$ is a 3×3 symmetric positive definite tensor prescribing element sizes, anisotropy and orientations to the mesh generator. $Q_{\mathcal{M}}(K) = 1$ corresponds to a perfectly regular element and $Q_{\mathcal{M}}(K) < 2$ correspond to excellent quality elements, while a high value of $Q_{\mathcal{M}}(K)$ indicates a nearly degenerated element. For non-adapted meshes, the identity matrix $\mathcal{I}_3$ is chosen as metric tensor.

The first mesh optimization tool is vertex smoothing which consists of relocating each vertex inside its ball of elements, *i.e.,* the set of elements having P_i as vertex. For each tetrahedron K_j of the ball of P_i, a new optimal position P_j^{opt} for P_i can be proposed to form a regular tetrahedron in metric space. The final optimal position P_i^{opt} is computed as a weighted average of all these optimal positions. This way, an element of the ball is all the more dominant if its quality in the original mesh is bad.

The second mesh optimization tool to improve mesh quality is generalized swapping/local-reconnection. Let α and β be the tetrahedra's vertices opposite to the common face $P_1P_2P_3$. Face swapping consists of suppressing this face and creating the edge $\mathbf{e} = \alpha\beta$. In this case, the two original tetrahedra are deleted and three new tetrahedra are created. A generalization of this operation exists and consists in reconnecting the inside of shells of tetrahedra [1,12,15]. The different edge swaps are generally denoted $n \to m$ where m is the number of new tetrahedra. In this work, edge swaps $3 \to 2$, $4 \to 4$, $5 \to 6$, $6 \to 8$ and $7 \to 10$ have been implemented.

2.4. Moving mesh algorithm

The changing-connectivity moving mesh algorithm (MMA) is described in Algorithm 1, where the different phases described above are put together.

Algorithm 1 Changing-Connectivity Moving Mesh Algorithm

While ($t < T^{end}$)

1. Solve mesh deformation: compute vertices trajectories
   (a) $\{\mathbf{d}^{body}_{|\partial\Omega_h}(t + \frac{\Delta t}{2})\}$ = Compute body vertex displacement from current translation speed $\mathbf{v}^{body}$, rotation speed $\boldsymbol{\theta}^{body}$ and acceleration $\mathbf{a}^{body}$ for $[t, t + \frac{\Delta t}{2}]$
   $\mathbf{d}(t + \frac{\Delta t}{2})$ = Solve mesh deformation problem $(\mathbf{d}^{body}_{|\partial\Omega_h}(t + \frac{\Delta t}{2}),\ \frac{\Delta t}{2})$
   (b) $\{\mathbf{d}^{body}_{|\partial\Omega_h}(t + \Delta t)\}$ = Compute body vertex displacement from current translation speed $\mathbf{v}^{body}$, rotation speed $\boldsymbol{\theta}^{body}$ and acceleration $\mathbf{a}^{body}$ for $[t, t + \Delta t]$
   $\mathbf{d}(t + \Delta t)$ = Solve mesh deformation problem $(\mathbf{d}^{body}_{|\partial\Omega_h}(t + \Delta t),\ \Delta t)$
   (c) $\{\mathbf{v}, \mathbf{a}\}$ = Deduce inner vertex speed and acceleration from both displacements $\left\{\mathbf{d}(t + \frac{\Delta t}{2}), \mathbf{d}(t + \Delta t)\right\}$
   (d) If predicted mesh motion is invalid then $\Delta t = \Delta t/2^n$ and goto 1.
   Else $T^{els} = t + \Delta t$
2. Moving mesh stage with mesh optimizations
   While ($t < T^{els}$)
   (a) δt = Get moving mesh time step $\left(\mathcal{H}^k, \mathbf{v},\ CFL^{geom}\right)$
   (b) $\mathcal{H}^k$ = Swaps optimization $\left(\mathcal{H}^k, Q^{swap}_{target}\right)$
   (c) $\mathbf{v}^{opt}$ = Vertices smoothing $\left(\mathcal{H}^k, Q^{smoothing}_{target},\ Q_{max}\right)$
   (d) $\mathcal{H}^{k+1}$ = Move the mesh and update vertices speed $\left(\mathcal{H}^k, \delta t, \mathbf{v}, \mathbf{v}^{opt}, \mathbf{a}\right)$
   (e) Check mesh quality: stop if too distorted.
   (f) $t = t + \delta t$

   EndWhile

EndWhile

In this algorithm, two time steps appear: a large one Δt for the mesh deformation computation, and a smaller one δt^{opt} corresponding to the steps where the mesh is optimized. The first one is set manually at the beginning of the computation, but can be automatically reduced if the mesh quality degrades. The second one is computed automatically, using the CFL^{geom} parameter as described below.

A good restriction to be imposed on the mesh movement is that vertices cannot cross too many elements on a single move between two mesh optimizations. Therefore, a geometric parameter CFL^{geom} is introduced to control the number of stages used to perform the mesh displacement between t and $t + \Delta t$. If CFL^{geom} is greater than one, the mesh is authorized to cross more than one element in a single move. The moving geometric time step is given by:

$$\delta t = CFL^{geom} \max_{P_i} \frac{h(\mathbf{x}_i)}{\mathbf{v}(\mathbf{x}_i)}, \tag{2}$$

where $h(\mathbf{x}_i)$ is the smallest altitude of all the elements in the ball of vertex P_i, and $\mathbf{v}(\mathbf{x}_i)$ its velocity.

After each mesh deformation solution, the quality of the mesh in the future is analyzed: if the quality is too low, the mesh deformation problem is solved again with a smaller time step.

3. Comparison of two mesh deformation methods

3.1. Elasticity-based PDE

The first method to compute the displacement of all vertices, *i.e.,* the mesh deformation, is a PDE-based method using a linear-elasticity analogy [1,4].More precisely, the inner vertices movement is obtained by solving an *elasticity-*

like equation with a $\mathbb{P}_1$ Finite Element Method (FEM):

$$\operatorname{div}(\sigma(\mathcal{E})) = 0\,, \quad \text{with} \quad \mathcal{E} = \frac{\nabla\mathbf{d} + {}^T\nabla\mathbf{d}}{2}\,, \tag{3}$$

where σ and $\mathcal{E}$ are respectively the Cauchy stress and strain tensors, and $\mathbf{d}$ is the Lagrangian displacement of the vertices. The Cauchy stress tensor follows Hooke's law for isotropic homogeneous medium , where ν is the Poisson ratio, E the Young modulus of the material and λ, μ are the Lamé coefficients:

$$\sigma(\mathcal{E}) = \lambda \operatorname{trace}(\mathcal{E})\, \mathcal{I}_d + 2\,\mu\,\mathcal{E} \qquad \text{or} \qquad \mathcal{E}(\sigma) = \frac{1+\nu}{E}\sigma - \frac{\nu}{E}\operatorname{trace}(\sigma)\, \mathcal{I}_d\,.$$

In our context, ν is typically chosen of the order of 0.48. This corresponds to a very soft, nearly incompressible material. Actually, this value for ν corresponds to a nearly ill-posed problem. Note that the closer ν is to 0.5, the harder it is to converge the Finite-Element linear system. The chosen value appears to be a good trade-off between material softness and the preservation of the iterative linear system solving algorithm efficiency. The FEM system is then solved by a Conjugate Gradient algorithm coupled with an LU-SGS pre-conditioner.

Two kinds of boundary conditions are considered. Dirichlet conditions are used to enforce strongly the displacement of the vertices located on moving boundaries. They are also used for fixed boundaries far from the moving bodies. These Dirichlet conditions can be relaxed if we want some vertices to move on a specific plane, typically on faces of a surrounding box close to the moving bodies, or even intersecting one (in the case of symmetry planes). In that case, if the plane is orthogonal to an axis of the Cartesian frame, only the displacement along this axis is enforced as a Dirichlet condition. The displacement in the two other directions are considered as degrees of freedom and is added in the FEM matrix as for normal volume points.

An advantage of elasticity-based methods is the opportunity they offer to adapt the local material properties of the mesh, especially its stiffness, according to the distortion and efforts born by each element: the way the Jacobian of the transformation from the reference element to the current element is accounted for in the FEM matrix assembly is modified. The classical $\mathbb{P}_1$ FEM formulation of the linear elasticity matrix leads to the evaluation of quantities of the form:

$$\int_K s\,\frac{\partial\varphi_J}{\partial x_k}\,\frac{\partial\varphi_I}{\partial x_l}\,\mathrm{d}\mathbf{x} = s\,|K|\,\frac{\partial\varphi_J}{\partial x_k}\,\frac{\partial\varphi_I}{\partial x_l}\,,$$

where s is either λ, μ or $\lambda + 2\mu$ and $|K|$ is the volume of tetrahedron K. The above quantity is replaced by:

$$\int_K s\left(\frac{|\hat{K}|}{|K|}\right)^{\chi}\frac{\partial\varphi_J}{\partial x_k}\,\frac{\partial\varphi_I}{\partial x_l}\,\mathrm{d}\mathbf{x} = s\,|K|\left(\frac{|\hat{K}|}{|K|}\right)^{\chi}\frac{\partial\varphi_J}{\partial x_k}\,\frac{\partial\varphi_I}{\partial x_l}\,,$$

where $\chi > 0$ is the stiffening power and $\hat{K}$ is the reference element. This technique locally multiplies λ and μ by a factor proportional to $|K|^{-\chi}$. χ determines how stiffer than large elements small elements are. In this work, we chose $\chi = 1$.

The main drawback of this method is that it is difficult to parallelize efficiently because of the pre-conditioner. Moreover, the system can be slow to converge if the system is too stiff.

3.2. Inverse Weighted Distance method

An alternative approach to computing the displacement of the volume vertices is by means of an interpolation approach. In the interpolation approach the displacement is defined by an algebraic interpolation function which provides a smooth function to blend displacements between boundary surfaces. Algebraic methods may either utilize implicit formulations such as those used in the popular Radial Basis Function (RBF) interpolation method [11], or explicit formulations such as transfinite interpolation [14] used for structured mesh generation. Generally explicit interpolation functions have performance advantages since they avoid problems associated with solving large stiff linear systems, however care must be taken to design interpolation functions that can sustain large deformations without introducing folded volume cells. For this evaluation we use the robust and fast explicit deformation method [19] that is an algebraic technique that utilizes a reciprocal distance weighted sum of deformation functions to interpolate deformations to the volume vertices. In this algebraic deformation method it is assumed that every vertex of the boundary

surfaces have a displacement field that can be approximated by a rigid body motion (displacement plus rotation about the node). This rigid body motion is computed from the displacements of neighbor vertices by solving a least squares problem fitting a quaternion to the relative displacements of neighboring vertices. The nodal displacement field in the neighborhood of vertex i can then be represented by the function denoted $\vec{s}_i(\vec{r})$, that is written as

$$\vec{s}_i(\vec{r}) = M_i\vec{r} + \vec{b}_i - \vec{r}, \tag{4}$$

where M_i is a rotation matrix, $\vec{b}_i$ is a displacement vector associated with the i^{th} node, and $\vec{r}$ is a coordinate vector in the original mesh. The displacement field in the volume mesh is then described through a weighted average of all boundary node displacement fields as given by

$$\vec{s}(\vec{r}) = \frac{\sum w_i(\vec{r})\vec{s}_i(\vec{r})}{\sum w_i(\vec{r})}. \tag{5}$$

The weighting function used to blend these displacements is based on a two-term inverse distance weighting function that is designed to preserve near-boundary mesh orthogonality while providing a smooth transition to blend between more distant surfaces. The nodal weight employed is defined by the function

$$w_i(\vec{r}) = A_i * \left[\left(\frac{L_{def}}{\|\vec{r} - \vec{r}_i\|}\right)^a + \left(\frac{\alpha L_{def}}{\|\vec{r} - \vec{r}_i\|}\right)^b\right], \tag{6}$$

where $\vec{r}_i$ is the position of point i, A_i is the area weight assigned to node i, L_{def} is an estimated length of the deformation region, α is an estimated fraction of L_{def} that is reserved for stiffer near body deformation, and a and b are user-defined exponents where a controls the smooth blending of deformations between surfaces while b controls the more stiff deformation close to deforming surfaces. Numerical experimentation suggests that $a = 3$ and $b = 5$ provide the best-quality for three-dimensional cases. The α parameter can be a fixed parameter or it can be dynamically computed based on the amount of deformation in the problem. Further details are described in a recent paper [19] and not discussed here.

Note, a naive implementation of the algorithm directly from Equation (5) will result in an $O(n^2)$ algorithm that will be too slow for practical applications. However, the localization provided by the reciprocal weights provides a mechanism for approximating the contributions of more distant points. Therefore the displacement function can be evaluated with a suitable tolerance utilizing a fast $O(n \log n)$ tree-code based algorithm that approximates the contributions of collections of distant surface nodes using a multi-pole expansion technique (as described in detail in [19]). When the algorithm is optimized using these techniques it can be very fast and highly parallelizable.

This algorithm has the advantage that it can be applied to meshes with arbitrary connectivity including adapted meshes with hanging nodes without difficulty. It is able to handle highly anisotropic mesh elements near viscous walls without difficulty and is free of numerical stiffness issues that can degrade the robustness of implicit methods for mapping deformations. However, since it is a global method that is based on surface mesh deformations only, it lacks an ability to dynamically adapt deformation to mesh conditions. For example, it is unable to adjust deformations locally to adapt to local mesh resolution requirements as can be easily accomplished with adjustments to element stiffness in PDE-based method.

3.3. Numerical Examples

We now study on challenging numerical examples how those two mesh deformation methods behave when used in the changing-connectivity moving mesh algorithm. The next examples are purely moving mesh simulations and we will focus our analysis on mesh quality criteria. In particular, we show that the MMA applies successfully to rigid body with large shearing and deformable bodies and we give an insight into a strategy to move boundary layers on deformable bodies.

3.3.1. Interpenetrating cylinders

The first example example is a challenging academic test case with rigid bodies, and has been presented in [1]. Two cylinders interpenetrate each other and there is only one layer of elements between both cylinders, *i.e.* internal edges

Table 1. Interpenetrating cylinders test case. Total number of mesh deformation solutions and mesh quality optimization steps to achieve the displacement, final mesh statistics and total number of swaps with the changing-connectivity MMA.

	# mesh deform.	# mesh optim.	Q_{end}^{mean}	$1 < Q_{end} < 2$	$Q_{overall}^{worst}$	# swaps
Elasticity	50	429	1.5	97.1%	79	852, 514
IDW	50	570	1.4	98.2%	30	1, 408, 680

connect both cylinders. The geometry of the domain is shown in Figure 1. To make this test case more complicated, we add rotation to the top cylinder:

$$\theta^{top} = (0, 0, 5)\,, \mathbf{v}^{top} = (0, 0, -0.1) \quad \text{and} \quad \theta^{bottom} = (0, 0, 0)\,, \mathbf{v}^{bottom} = (0, 0, 0.1)\,.$$

The initial mesh is composed of 34 567 vertices and 188 847 tetrahedra.

A shear layer appears between the two cylinders, that is only one element wide. Therefore, the swapping optimization has just a little of freedom to act. Moreover, it is obvious that too large Δt will lead to an invalid mesh: such a simulation requires a relatively large number of mesh deformation solutions. For both mesh deformation methods, 50 mesh deformation steps are set, *i.e.* $\Delta t = 1$, and we set $CFL^{geom} = 2$.

The changing-connectivity MMA proves to be extremely robust in both cases, keeping the mesh worst element quality below 100 and ensuring a final excellent mean quality of 1.5 . Because of the shear, a large number of swaps (around 1 million) have been performed. For that test case, results with both mesh deformation methods are very similar. Moreover, Figure 1 (center) points out a major difference between elasticity and IDW: elasticity creates rotational displacements in the mesh above and under the cylinder, that push vertices in front, attract vertices behind the moving inner cylinder, and thus move vertices to empty areas instead of crushing them, while the IDW method tends to produce straighter displacements. Table 1 provides a comparison of the MMA with the two mesh deformation methods.

3.3.2. Squeezing can

The next test case deals with deformable geometries. It represents a can which is squeezed. The can is a cylinder of length 5 units and diameter 2 units inside a spherical domain of radius 10 units. The can is centered at the origin and is aligned along the z-axis. The simulation runs up to $T^{end} = 1$ and the temporal deformation function is given by:

$$\text{if } |z(0)| < 2 \text{ then } \mathbf{x}(t) = \begin{cases} x(t) = x(0) \\ y(t) = \left(\alpha + (1-\alpha)(1-t^2)\right) y(0) \\ z(t) = z(0) \end{cases} \qquad \text{with} \qquad \alpha = \frac{1}{2.2}\left(1.2 - \cos\left(\frac{\pi}{2}|z(0)|\right)\right).$$

The can deformation is shown in Figure 2. For this case, the moving mesh simulation is done for the inside mesh of the can geometry. The mesh is composed of 7 018 vertices and 36 407 tetrahedra. Four mesh deformation steps are initially prescribed, and CFL^{geom} is fixed to 1. Statistics for the case are given in Table 2.

The mesh is successfully moved,as shown on Figure 2. We clearly see that the elasticity methods has moved vertices away from the squeezed region while keeping an excellent mesh quality, whereas the IDW tends to let them crush. For this case, the swap optimization are essential to preserve the quality of the mesh all along the movement.

3.3.3. Bending beam

This test case has been proposed in [19]. It is a general deformation test on a three-dimensional bending beam. The beam is 8 units long, 2 units wide and 0.1 unit thick inside a spherical domain of radius 25 units. The beam is deformed such that the beam center-line is mapped onto a circular arc:

$$\mathbf{x}(t) = \begin{cases} x(t) = (R - z(0))\sin(\theta) \\ y(t) = y(0) \\ z(t) = z(0) + (R - z(0))(1 - \cos(\theta)) \end{cases} \qquad \text{with } \theta = \frac{x(0)}{R} \quad \text{and } R = R_{\min} + (R_{\max} - R_{\min})\frac{\mathrm{e}^{-\kappa t} - \mathrm{e}^{-\kappa}}{1 - \mathrm{e}^{-\kappa}}\,.$$

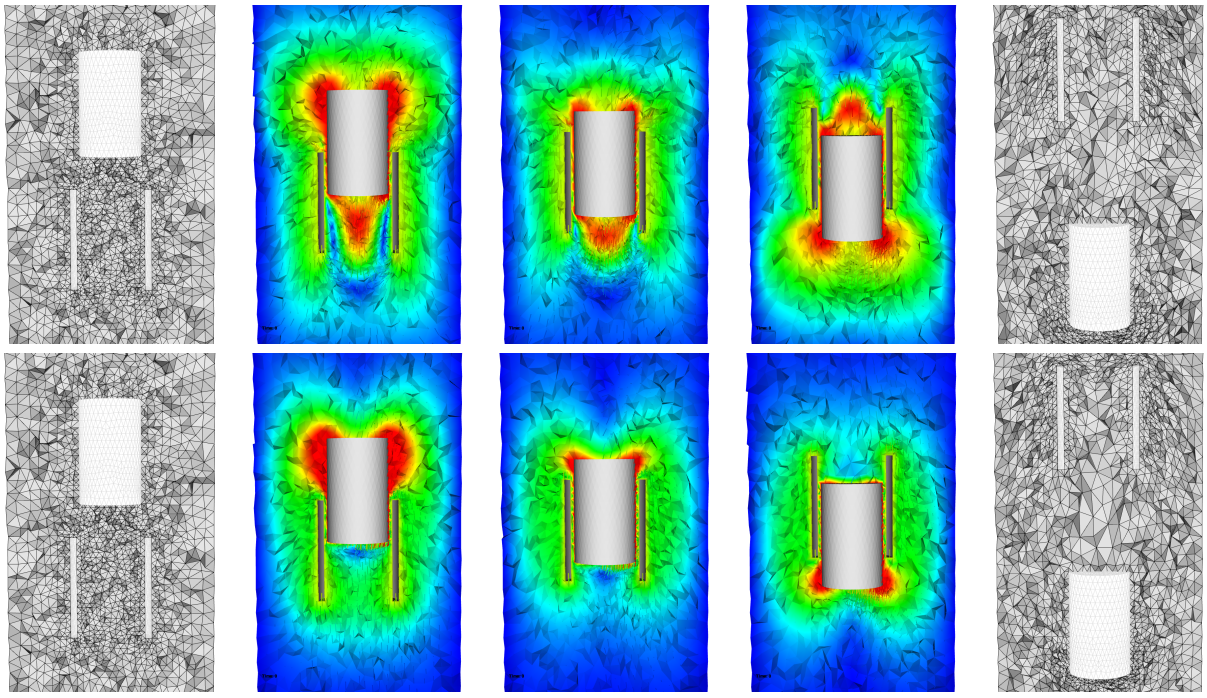

Fig. 1. Interpenetrating cylinders test case. Evolution of the mesh with the elasticity-based (top) and IDW (bottom) mesh deformation methods. Between the initial (left) and final (right) meshes, the displacements given by the mesh deformation steps at times 12, 18 and 25 is shown.

Table 2. Squeezing can test case. Total number of mesh deformation solutions and mesh quality optimization steps to achieve the displacement, final mesh statistics and total number of swaps with the changing-connectivity MMA.

	# mesh deform.	# mesh optim.	Q_{end}^{mean}	$1 < Q_{end} < 2$	$Q_{overall}^{worst}$	# swaps
Elasticity (in.)	6	27	1.5	93.0%	11	16.255
IDW (in.)	4	27	1.48	92.5%	11	11,940

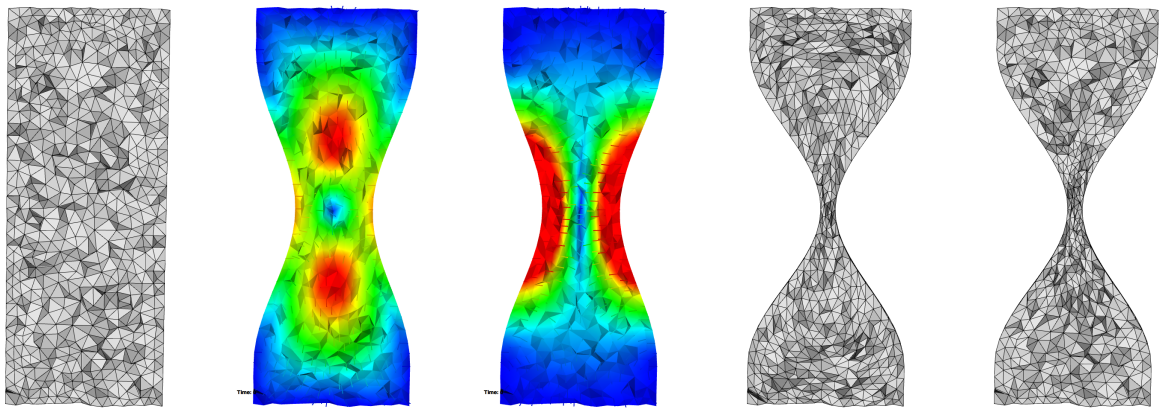

Fig. 2. Squeezing can test cases. From left to right: initial mesh, displacement given by the elasticity-based and IDW mesh deformation problems at times 0.75, and final meshes with elasticity and IDW.

where $R_{\min}$ and $R_{\max}$ are the radii of curvature at $t = 0$ and $t = 1$, respectively. κ is a parameter that controls how rapidly the radius and center of curvature move from the initial to final values. For this test case, $R_{\min} = 2$, $R_{\max} = 500$ and $\kappa = 10$. The initial mesh is composed of 58 315 vertices and 322 971 tetrahedra. We set $\Delta t = 0.2$ and $CFL^{geom} = 1$.

Again, this large deformation problem is solved without any difficulty, requiring only a few mesh deformation steps, the final mesh is excellent as shown in Table 3 and no skewed element appears inside the mesh. There again, we can see in Figure 3 that the elasticity-based method tends to move points farther from the body than the IDW method, thus requiring significantly more swaps. However, the performances of the elasticity-based and IDW methods are very close, cf. Table 3.

Table 3. Bending beam test case. Total number of mesh deformation solutions and mesh quality optimization steps to achieve the displacement, final mesh statistics and total number of swaps with the changing-connectivity MMA.

	# mesh deform.	# mesh optim.	Q^{mean}_{end}	$1 < Q_{end} < 2$	$Q^{worst}_{overall}$	# swaps
Elasticity	5	211	1.3	99.9%	4.4	479, 182
IDW	4	40	1.3	99.8%	5.9	149, 536

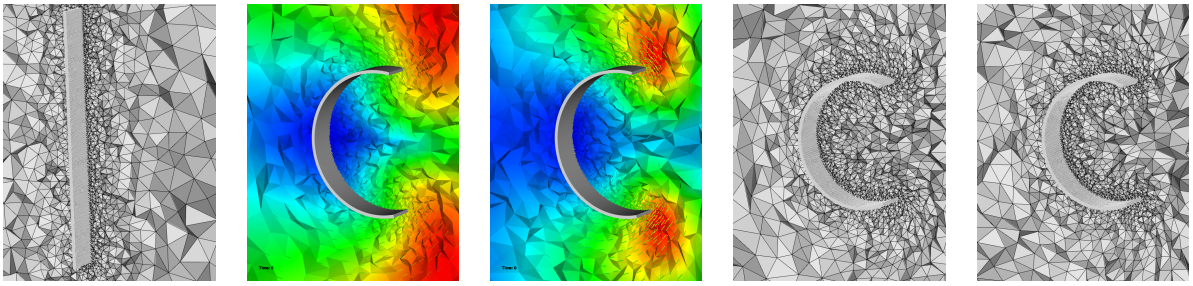

Fig. 3. Bending beam test case. From left to right: initial mesh, displacement given by the elasticity-based and IDW mesh deformation problems at time 0.8, and final meshes with elasticity and IDW.

3.3.4. Boundary layers on deformable geometries

Dealing with boundary layers (BL) meshes for rigid bodies is rather easy: the whole mesh layers are rigidified and moved together with the body. However, it is a lot more difficult with deformable bodies, since the mesh structured layers must both follow the deformation of the body and keep their structure. It was *a priori* not certain that the movement provided by the elasticity or IDW step would be precise enough to keep that structure. In the sequel we demonstrate that both methods can move one thick layer with a good accuracy, and that such layers can also be moved correctly with the elasticity method.

One boundary layer. For this case, we consider the bending beam studied previously, to which we add one thin structured mesh layer of height a quarter of the height of the beam, see Figure 4. The movement of the beam is the same and no swap or smoothing optimizations are done in the BL. At the end of the simulation, we observe that the structured aspect of the BL has been well preserved. In Table 4, we gather quality indicators depending on different moving mesh parameters. We consider the variation of the distance of the layer vertices to the body, that is expected to remain constant in ideal cases. We can see that taking acceleration into account in the trajectories is very important: indeed, as the movement of the beam is accelerated, the layer under the beam tends to thicken while the layer above it tends to be crushed. We can also see that less IDW steps are required to reach an equivalent mean quality, but the maximal distance variation is less good.

This result is very promising, because the structured aspect of the BL mesh is well preserved. This lets us consider a clever strategy to move BL meshes with deformable geometries.

Table 4. Bending beam with one BL. Mean and maximal relative distance variations $\delta \mathrm{d}x$, in the case of 10 elasticity solutions, 50 elasticity solutions with linear (lin.) trajectories (without taking acceleration into account), 50 elasticity solutions with quadratic (quad.) trajectories.

	10 elast. quad.	50 elast. lin.	50 elast. quad.	10 IDW quad.
Mean relative $\delta \mathrm{d}x$ (in %)	27	9.9	5.7	6.4
Max relative $\delta \mathrm{d}x$ (in %)	143	66	19	60

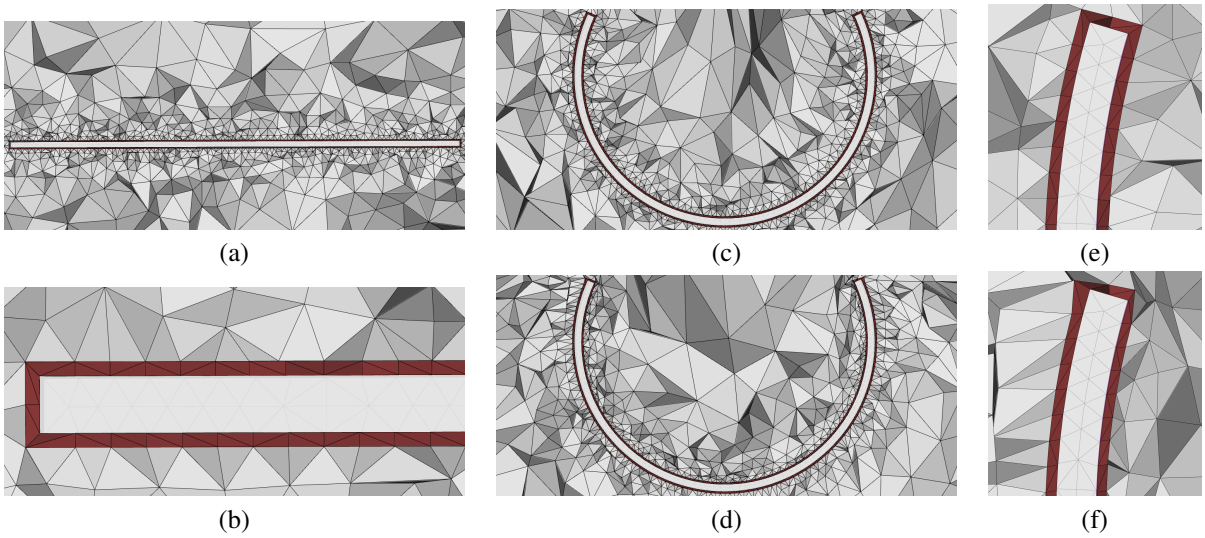

Fig. 4. One BL test case. Initial mesh (a) and close up on the BL (b), final mesh with elasticity (c) and IDW (d), and close-ups on the final BL meshes with elasticity (e) and IDW (f).

Several boundary layers. The elasticity mesh deformation method is also actually capable of moving several thinner BL, provided the mesh deformation time step is small enough to handle the displacement of the smallest elements against the body. The simulation was run with a boundary layer containing 10 thin layers (the smallest height is 0.0017 units): the structure inside the layer is still well preserved all along the movement, and the quality of the layer looks good even at the end of the deformation of the beam. Close-ups on the BL meshes are show in Figure 5. We were not able to move these layers with the IDW mesh deformation, which needs to be further investigated.

4. Moving-mesh ALE computations

The changing-connectivity moving mesh algorithm described above is coupled to our in-house flow solver to run CFD simulations with moving geometries. The compressible Euler equations are solved with a Finite Volume method. The displacements of the vertices require the use of the ALE formulation of these equations. As regards the temporal accuracy, the considered SSPRK schemes are based on the strict application of the Discrete Geometrical Conservation Law (DGCL). Details about the ALE solver are given in [5]. The absence of remeshing preserves the quality of the solution, as it is free of most interpolation errors due to solutions transfers from one mesh to another.

The flow solver is integrated to Algorithm 1 as follows. The solver iterations are embedded in the optimization phase, so that the mesh is moved after each solver iteration, but optimizations are performed only when the optimization time (defined by the CFL^{geom} parameter) is reached. If necessary, the solver time step is truncated to perform optimizations at the correct time.

The efficiency of the method is demonstrated on two examples involving complex moving geometries. Both simulations were run using the IDW interpolation and the elasticity-based mesh deformation methods.

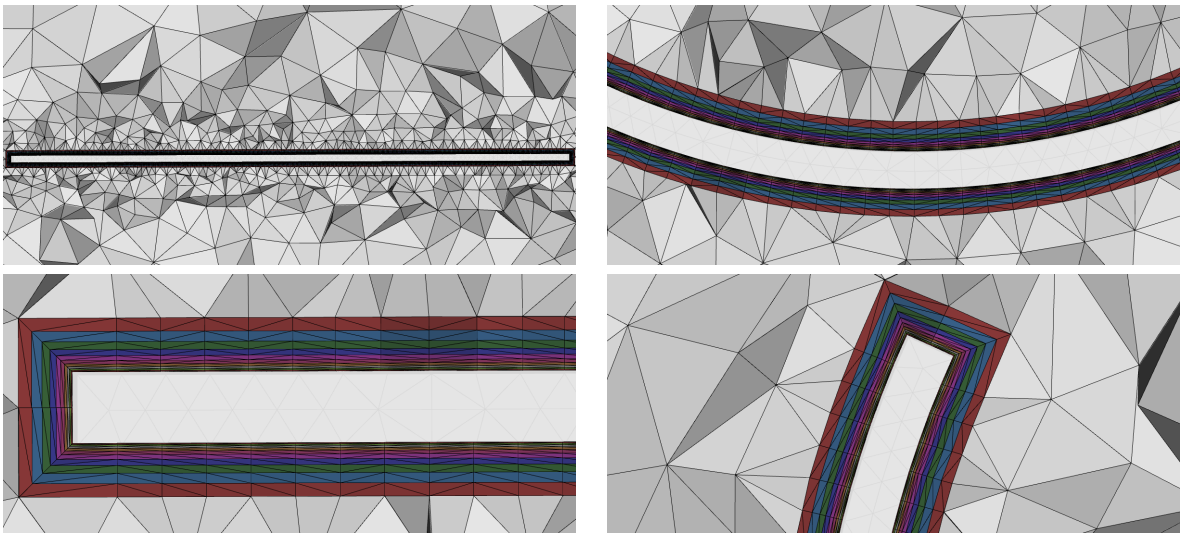

Fig. 5. Ten BL test case. Initial mesh (left) and close-ups on the final BL mesh (right), with the elasticity-based mesh deformation method.

4.1. Vortical wake of a F117 aircraft nosing up

The second example is a subsonic F117 aircraft nosing up, that creates a vortical wake. An inflow of air at Mach 0.4 arrives in front of the aircraft, initially in horizontal position that noses up, stays up for a while, then noses down. In this example, the aicraft rotates around its center of gravity. Let $T = 1s$ be the characteristic time of the movement and $\theta^{max} = 20^o$ the maximal angle reached, the movement is defined by its angle of rotation, of which the evolution is divided in 7 phases. Phase (i) ($0 \leq t \leq T/2$) is an initialization phase, during which the flow around the aircraft is established. Phase (ii) ($\frac{T}{2} < t \leq T$) and (iii) ($T < t \leq \frac{3T}{2}$) are respectively phases of accelerated and decelerated ascension. Vortices start to grow behind the aircraft, and they expand during phase (iv) ($\frac{3T}{2} < t \leq \frac{7T}{2}$), where the aircraft stays in upward position. Phase (v) ($\frac{7T}{2} < t \leq 4T$) and (vi) ($4T < t \leq \frac{9T}{2}$) are phases of accelerated and decelerated descent, the vortices start to move away and they slowly disappear in phase (vii) ($\frac{9T}{2}t \leq 5T$). Wall conditions are imposed on the faces of the surrounding box and slipping conditions on the aircraft.

The initial mesh has $501,859$ vertices and $3,012,099$ tetrahedra. The mesh used and snapshots of the solutions are shown in Figures 6 and 7. A difficulty of this test case is the uneven movement of the aircraft, with strong acceleration and deceleration during the phases of ascension and descent, and the inversion of the acceleration in the middle of these phases. It is necessary to solve enough mesh deformation problems to avoid that the body boundaries cross small elements close to it. Statistics of the moving mesh characteristics are show in Table 5. We can see that the two proposed mesh deformation methods produce rather similar results in terms of mesh quality, but the case requires only half as many mesh deformation steps if the IDW method is used.

Table 5. Nosing up F117 test case. Total number of mesh deformation solutions and mesh quality optimization steps to achieve the displacement, final mesh statistics and total number of swaps with the changing-connectivity MMA.

	# mesh deform.	# mesh optim.	Q_{end}^{mean}	$1 < Q_{end} < 2$	$Q_{overall}^{worst}$	# swaps
Elasticity	24	66	1.4	99.8%	19	698 755
IDW	12	37	1.4	99.8%	19	713 247

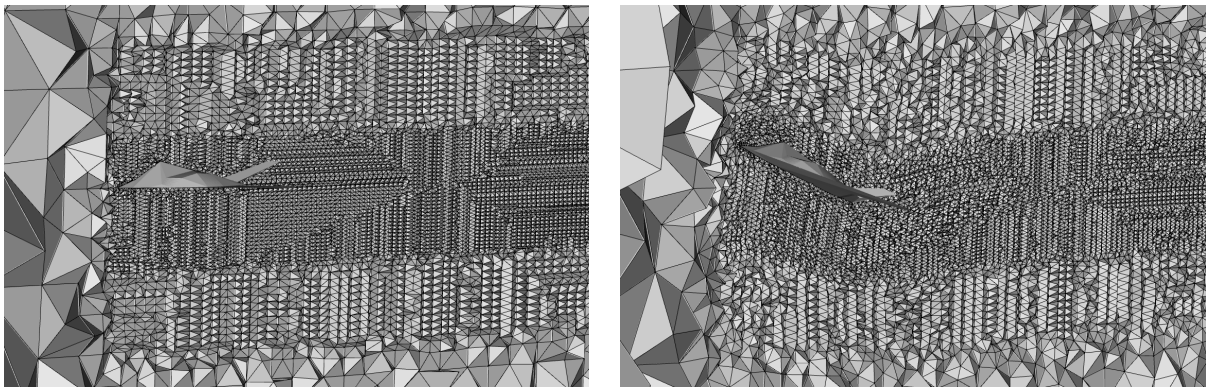

Fig. 6. Test case of the nosing up F117. semi-structured meshes in initial horizontal position, and upward position.

4.2. Two F117 aircraft flight paths crossing

The first example is modeling two F117 aircrafts having crossing flight paths. This problem illustrates the efficiency of the connectivity-change moving mesh algorithm in handling large displacements of complex geometries without any remeshing. When both aircrafts cross each other, the mesh deformation encounters a large shearing due to the opposite flight directions. The connectivity-change mesh deformation algorithm handles easily this complex displacement thanks to the mesh local reconnection. Therefore, the mesh quality remains very good during the whole displacement.

As concerns the fluid simulation, the aircrafts are moved at a speed of Mach 0.8, in an initially inert uniform fluid: at $t = 0$ the speed of the air is null everywhere. The rotation speed of the aircrafts is set to a tenth of their translation speed. Transmitting boundary conditions are used on the sides of the surrounding box, and slipping conditions are imposed on the two F117 bodies. After a phase of initialization, the flow is established when the two F117s pass each other.

The initial mesh has $586,610$ vertices and $3,420,332$ tetrahedra. Some statistics on the moving mesh aspect of the simulation are shown in Table 6. We can see that both methods keep an excellent mesh quality all along the simulation. We used the minimal number of mesh deformations possible, and the IDW method again requires half as many mesh deformation steps, but then requires almost five times more swaps for the whole simulation. This is due to the fact that with the elasticty-based approach, vertices tend to bypass the moving bodies, while their trajectories are straighter with the IDW, thus creating a lot of shearing. In Figure 8, we show both the moving mesh aspect of the simulation and the flow solution at different time steps.

Table 6. Two F117s test case. Total number of mesh deformation solutions and mesh quality optimization steps to achieve the displacement, final mesh statistics and total number of swaps with the changing-connectivity MMA.

	# mesh deform.	# mesh optim.	Q_{end}^{mean}	$1 < Q_{end} < 2$	$Q_{overall}^{worst}$	# swaps
Elasticity	43	1563	1.4	99.8%	27	$2,176,850$
IDW	22	1717	1.4	99.7%	26	$10,396,127$

5. Conclusion

In this paper, new numerical evidence has been given that 3D large displacements are possible with a mesh deformation algorithm coupled with mesh optimization using only swaps and vertex movements. No re-meshing is required. The possibility to compute the mesh deformation using an explicit Inverse Distance Weighted interpolation has been added to the algorithm. Moreover, we have been dealing with deformable geometries, and have started to

address the issue of boundary layers. Finally, the moving mesh algorithm has been coupled to a CFD ALE flow solver. Several examples, both purely moving-mesh and CFD, have been exhibited.

The comparison of the two mesh deformation methods shows that both are very robust coupled with mesh optimizations. They both produce meshes of equivalent excellent qualities.It seems that in general, less IDW solutions are necessary, but the elasticity-based approach requires significantly less swap in the cases with a lot of shearing. These results need to be confirmed and refined on more cases, but they confirm the advantage of having both tools at our disposal.

However some problematics remain. The moving of boundary layers need to be further developed. The implementation of the IDW method and the parallelization of the flow solver must be improved, in order to run CPU time comparisons of both methods. Other issues are surface optimizations, simulations of deformable geometries with possible connectivity changes and the efficient treatment of contact problems.

6. Acknowledgments

This work was partially funded by the Airbus Group Foundation.

References

[1] F. Alauzet. A changing-topology moving mesh technique for large displacements. *Engineering with Computers*, 30(2):175–200, 2014.

[2] F. Alauzet and A. Loseille. High-order sonic boom prediction by utilizing mesh adaptive methods. In *48th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA-2010-1390, Orlando, FL, USA, Jan 2010.

[3] F. Alauzet and G. Olivier. Extension of metric-based anisotropic mesh adaptation to time-dependent problems involving moving geometries. In *49th AIAA Aerospace Sciences Meeting*, AIAA Paper 2011-0896, Orlando, FL, USA, Jan 2011.

[4] T.J. Baker and P. Cavallo. Dynamic adaptation for deforming tetrahedral meshes. *AIAA Journal*, 19:2699–3253, 1999.

[5] N. Barral and F. Alauzet. Large displacement body-fitted FSI simulations using a mesh-connectivity-change moving mesh strategy. In *44th AIAA Fluid Dynamics Conference, AIAA-2014, Atlanta, GA, USA*, June 2014.

[6] J.D. Baum, R. Löhner, T.J Marquette, and H. Luo. Numerical simulation of aircraft canopy trajectory. In *35th AIAA Aerospace Sciences Meeting*, AIAA Paper 1997-0166, Reno, NV, USA, Jan 1997.

[7] J.D. Baum, H. Luo, and R. Löhner. A new ALE adaptive unstructured methodology for the simulation of moving bodies. In *32th AIAA Aerospace Sciences Meeting*, AIAA Paper 1994-0414, Reno, NV, USA, Jan 1994.

[8] J.A. Benek, P.G. Buning, and J.L. Steger. A 3D chimera grid embedding technique. In *7th AIAA Computational Fluid Dynamics Conference*, AIAA Paper 1985-1523, Cincinnati, OH, USA, Jul 1985.

[9] F. Brezzi, J.L. Lions, and O. Pironneau. Analysis of a Chimera method. *C.R. Acad. Sci. ParisSer. I*, 332(7):655–660, 2001.

[10] G. Compere, J-F. Remacle, J. Jansson, and J. Hoffman. A mesh adaptation framework for dealing with large deforming meshes. *Int. J. Numer. Meth. Engng*, 82(7):843–867, 2010.

[11] A. de Boer, M. van der Schoot, and H. Bijl. Mesh deformation based on radial basis function interpolation. *Comput. & Struct.*, 85:784–795, 2007.

[12] E. Brière de l'Isle and P.L. George. Optimization of tetrahedral meshes. *IMA Volumes in Mathematics and its Applications*, 75:97–128, 1995.

[13] C. Dobrzynski and P.J. Frey. Anisotropic Delaunay mesh adaptation for unsteady simulations. In *Proceedings of the 17th International Meshing Roundtable*, pages 177–194. Springer, 2008.

[14] L. E. Eriksson. Generation of boundary conforming grids around wing-body configurations using transfinite interpolation. *AIAA Journal*, 20:1313–1320, 1982.

[15] P.J. Frey and P.L. George. *Mesh generation. Application to finite elements*. ISTE Ltd and John Wiley & Sons, 2nd edition, 2008.

[16] P.L. George. Tet meshing: construction, optimization and adaptation. In *Proceedings of the 8th International Meshing Roundtable*, South Lake Tao, CA, USA, 1999.

[17] O. Hassan, K.A. Sørensen, K. Morgan, and N. P. Weatherill. A method for time accurate turbulent compressible fluid flow simulation with moving boundary components employing local remeshing. *Int. J. Numer. Meth. Fluids*, 53(8):1243–1266, 2007.

[18] R. Löhner, J.D. Baum, E. Mestreau, D. Sharov, C. Charman, and D. Pelessone. Adaptive embedded unstructured grid methods. *Int. J. Numer. Meth. Engng*, 60:641–660, 2004.

[19] E. Luke, E. Collins, and E. Blades. A fast mesh deformation method using explicit interpolation. *J. Comp. Phys.*, 231:586–601, 2012.

[20] D. Marcum and N. Weatherill. Unstructured grid generation using iterative point insertion and local reconnection. *AIAA Journal*, 33(9):1619–1625, 1982.

[21] S. Murman, M. Aftosmis, and M. Berger. Simulation of 6-DOF motion with cartesian method. In *41th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA-2003-1246, Reno, NV, USA, Jan 2003.

[22] C.S. Peskin. Flow patterns around heart valves: a numerical method. *J. Comp. Phys.*, 10:252–271, 1972.

[23] M. L. Staten, S. J. Owen, S. M. Shontz, A. G. Salinger, and T. S. Coffey. A comparison of mesh morphing methods for 3D shape optimization. In *Proceedings of the 20th International Meshing Roundtable*, pages 293–310. Springer, 2011.

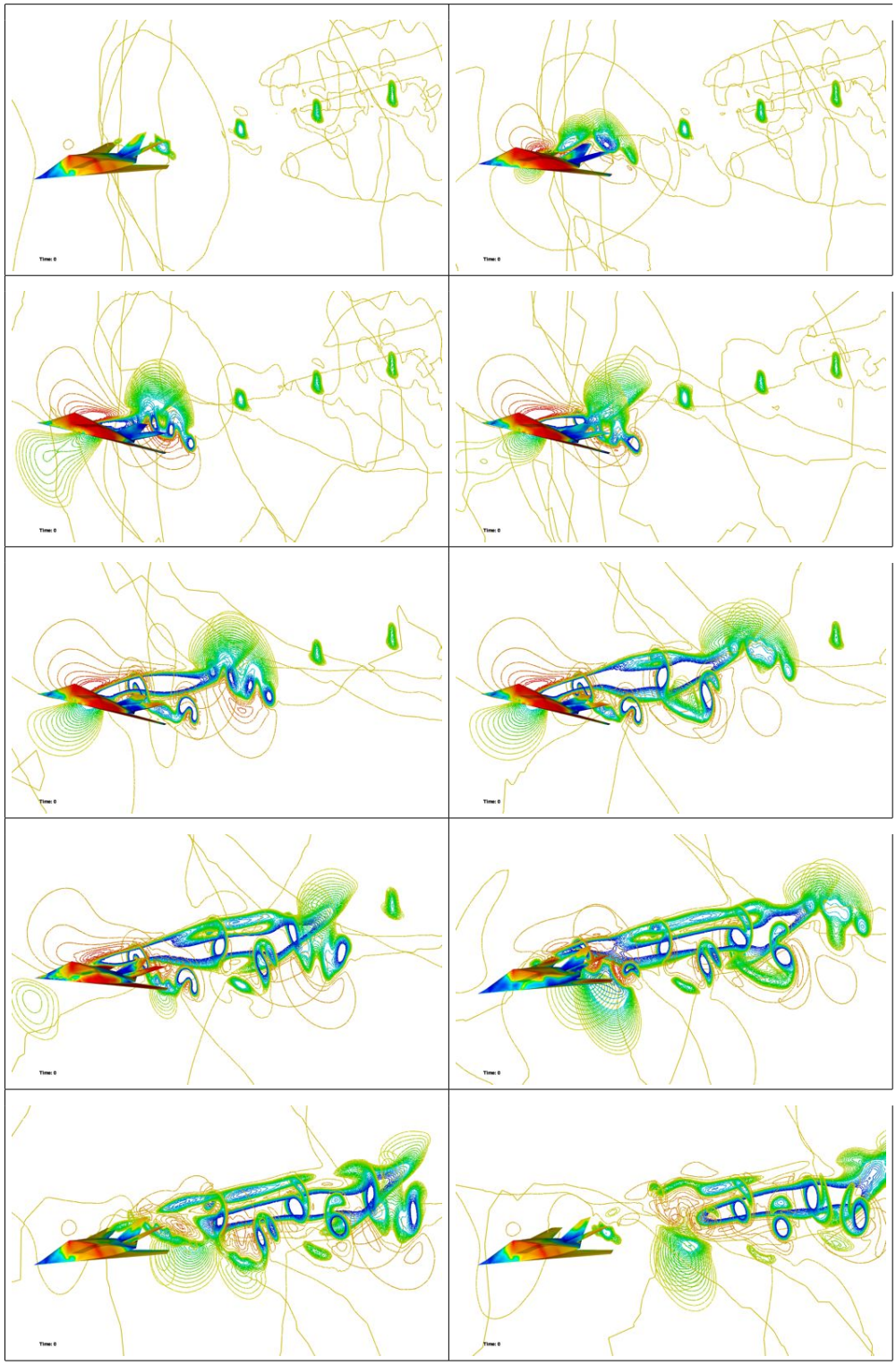

Fig. 7. Test case of the nosing up F117. Isolines on several cutting planes of the mach field at different time steps (from left to right and up to down t = 0.49, 0.85, 1.22, 1.47, 2.08, 2.82, 3.31, 3.80, 4.28 and 4.90).

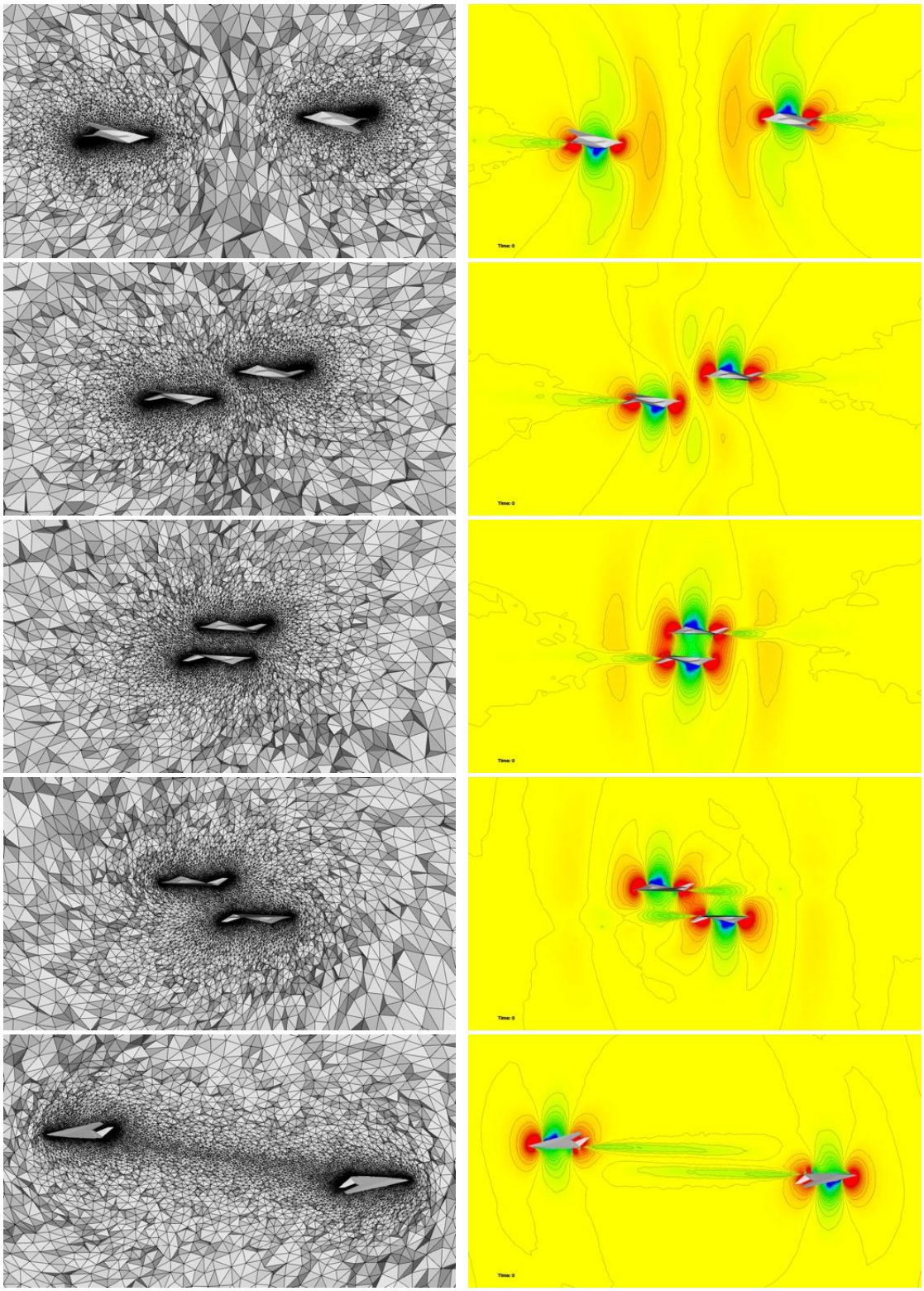

Fig. 8. Test case of the two F117s: Snapshots of the moving geometries and the mesh (left) and density (right).